# PK_GRIB2

## PACKS DATA INTO GRIB2 FORMAT

Bryon Lawrence
April 5, 2001
Bob Glahn
David Rudack
March 15, 2002

PURPOSE: To pack a gridded data field using the algorithm put forth in version two of the World Meteorological Organization's (WMO) standard for the exchange of General Regularly-distributed Information in Binary form (GRIB2). GRIB2 provides a method of compressing data without loss (depending on the scaling factors that the user chooses). The packed data are stored into a structured message that also contains information that identifies and defines the packed data grid. Such information includes the time of generation of the gridded product, the source of the gridded product, the type of map projection the gridded product uses, what the data in the gridded product represents, and which packing method was used to compress the data in the gridded product.

The user passes the gridded data field into this routine through either a two-dimensional integer or floating point array depending on the type of data contained in the data field. The information that identifies and defines the data field and the packing method used to compress it is passed into this routine through seven one-dimensional, integer "section" arrays. Each of these arrays corresponds to one of the first seven mandatory sections of the GRIB2 message (**Sections 0**, **1**, **3**, **4**, **5**, **6**, and **7**). Note that the user does not need to provide information for **Section 8** (the **End Section**). Supplemental data providing further qualification and quantification of a gridded product beyond what the structure of GRIB2 already provides for can be placed into **Section 2** (the **Local Use Section**). **Section 2** is optional and does not have to be included in the GRIB2 message.

This routine supports the simple, complex, and complex with second order spatial differences packing methods. If the simple packing method is chosen, then the data field may contain primary missing values. If the complex packing method is used, then the data field may contain primary and secondary missing values (secondary missing values may only be used if there can be primary missing values). The user has the option of passing in the data field with the primary missing values within it; or the user may pass in the data field with the primary missing values removed from it. In the latter case, the data field must be accompanied by a bit-map identifying the locations of the missing values relative to the defined grid. Note that these options do not exist with secondary missing values; they must always be supplied to this routine in their proper locations in the gridded data field.

More than one data grid may be packed into a GRIB2 message. This routine provides the functionality needed to pack multiple data grids into a single GRIB2 message. This is accomplished through repetitive calls to this routine with the "NEW" calling argument (see below) properly set. According to WMO GRIB2 regulations, **Sections 2** through **7**, **3** through **7**, or **4** through **7** may be repeated for each data grid packed into the GRIB2 message.

For a complete description of the GRIB2 format, templates, and code tables, the user is referred to the WMO document **FM 92-XII GRIB**.

This routine supports the packing of the following templates:

**<u>Grid Definition Templates (Section 3)</u>**

| | |
|---|---|
| **Template 3.0** | (Latitude/Longitude) |
| **Template 3.10** | (Mercator) |
| **Template 3.20** | (Polar Stereographic Projection) |
| **Template 3.30** | (Lambert Conformal) |
| **Template 3.90** | (Space View Perspective or Orthographic) |
| **Template 3.110** | (Equatorial Azimuthal Equidistant Projection) |
| **Template 3.120** | (Azimuth-range Projection) |

**<u>Product Definition Templates (Section 4)</u>**

| | |
|---|---|
| **Template 4.0** | (Analysis or Forecast at a Horizontal Level or in a Horizontal Layer at a Point in Time) |
| **Template 4.1** | (Individual Ensemble Forecast, Control and Perturbed, at a Horizontal Level or in a Horizontal Layer at a Point in Time) |
| **Template 4.2** | (Derived Forecast Based on All Ensemble Members at a Horizontal Level or in a Horizontal Layer at a Point in Time) |
| **Template 4.8** | (Average, Accumulation, and/or Extreme Values at a Horizontal Level or in a Horizontal Layer in a Continuous or Non-continuous Time Interval) |
| **Template 4.20** | (Radar Product) |

**Template 4.30** (Satellite Product)

**Data Representation Templates (Section 5)**

**Template 5.0** (Simple Packing)
**Template 5.2** (Complex Packing)
**Template 5.3** (Complex Packing and Spatial Differences) (Note that only second order spatial differences are supported at this time.)

**Data Templates (Section 7)**

**Template 7.0** (Grid Point Data - Simple Packing)
**Template 7.2** (Grid Point Data - Complex Packing)
**Template 7.3** (Grid Point Data - Complex Packing and Spatial Differences)

**CALL AND EXPLANATION OF FORMAL PARAMETERS :**

```
      CALL PK_GRIB2(KFILDO,AIN,IAIN,NX,NY,IDAT,NIDAT,
     1         RDAT,NRDAT,IS0,NS0,IS1,NS1,IS3,NS3,
     2         IS4,NS4,IS5,NS5,IS6,NS6,IS7,NS7,IB,
     3         IBITMAP,IPACK,ND5,MISSP,XMISSP,MISSS,
     4         XMISSS,NEW,MINPK,ICLEAN,L3264B,JER,
     5         NDJER,KJER)
```

KFILDO - Unit number of the output diagnostic (print) file. All lines of source that create diagnostic output in the packer routine are “commented out” with a “D” in column 1 of the source code. If the user desires that diagnostic information be generated when this packer is executed, then the option specific to the Fortran compiler being used that allows the compilation of debug lines as source code must be used when building the packer library. When diagnostic information is desired, the user must make sure that the file represented by this number has been opened **prior** to calling the “**pk_grib2**" routine. (INPUT)

AIN(L,M)- Array containing the gridded data to be packed (L=1,NX) (M=1,NY). This array is used only when the user is attempting to pack a data field containing floating point values. If the user wants to pack a data field consisting of integer values, then IAIN( ) (see below) must be used to pass the data into the packer. Note that when packing a floating point data field, the user must set element 21 in the IS5 array to “0". The contents of this array are not modified. (INPUT)

IAIN(L,M) Array containing the gridded data to be packed when the user is attempting to pack a data field containing integer values (L=1, NX) (M=1, NY). If the user wants to pack a data field consisting of floating point values, then AIN( ) (see above) must be used to pass the data field into the packer. Note that when packing an integer data field, the user must set element 21 in the IS5 array to “1". The contents of this array are not modified. (INPUT)

NX - The number of rows in the gridded product. (INPUT)

NY - The number of columns in the gridded product. (INPUT)

IDAT(L) - Contains the local use data consisting of a group or groups of integer values that will be packed into **Section 2**, the **Local Use Section**, of the GRIB2 message (L=1,NIDAT). See the special documentation concerning local use data below for an explanation of how the integer data must be formatted in this array. (INPUT)

NIDAT - The number of elements in the IDAT( ) array. (INPUT)

RDAT(L) - Contains the local use data consisting of a group or groups of floating point values that will be packed into **Section 2**, the **Local Use Section**, of the GRIB2 message (L=1,NRDAT). See the special documentation concerning local use data below for an explanation of how the floating point data must be formatted in this array. (INPUT)

NRDAT - The number of elements in the RDAT( ) array. (INPUT)

IS0(L) - Holds the data elements of **Section 0** (the **Indicator Section**) (L=1,NS0). The contents of this array are modified by this routine. The user only needs to supply a value in **element 7** of this array representing the discipline of the processed data in the GRIB2 message. See the GRIB2 section outline below for a description of the data elements contained in **Section 0**. (INPUT/OUTPUT)

NS0 - The dimension of IS0( ). NS0=16 is sufficient. (INPUT)

IS1(L) - Holds the data elements of **Section 1** (the **Identification Section**) (L=1,NS1). The contents of this array are modified by this routine. See the GRIB2 section outline below to determine which elements in this array the user needs to supply values for. (INPUT/OUTPUT)

NS1 - The dimension of IS1( ). NS1=21 is sufficient. (INPUT)

IS3(L) - Holds the values of **Section 3** (the **Grid Definition Section**) to be packed into the GRIB2 message (L=1,NS3). **Section 3** defines the type of map projection that the data are in reference to. The contents of this array are modified by this routine. See the GRIB2 section outline below to determine which array elements the user needs to supply values for. (INPUT/OUTPUT)

NS3 - The number of elements in the IS3( ) array. Since the grid definition templates are of variable size, the value of this parameter depends upon what type of map the data field being packed is projected on. NS3=96 is sufficient for all templates except template 3.120, the Azimuth-Range Projection, which is used for radar images. In the case where template 3.120 is being used, NS3=1600 should be sufficient. (INPUT)

IS4(L) - Holds the values of **Section 4** (the **Product Definition Section**) to be packed into the GRIB2 message (L=1,NS4). **Section 4** defines what the data in the data field being packed represent, i.e., does the data represent a map of 1-hour rainfall totals or does it represent the height contours on an Aviation Model 500-mb height forecast grid. The contents of this array are modified by this routine. See the GRIB2 section outline below to determine which array elements the user needs to supply values for. (INPUT/OUTPUT)

NS4 - The number of elements in the IS4( ) array. Since the product definition templates are of variable size, the value of this parameter depends upon what type of product the data field being packed represents. NS4=60 should be sufficient for all of the Section 4 product definition templates supported by this packer, with the possible exception of **template 4.30,** the **Satellite Product**, which could require more array space depending upon the number of contributing bands in the satellite image. (INPUT)

<u>IS5</u>(L) - Holds the data values of **Section 5** (the **Data Representation Section**) to be packed into the GRIB2 message (L=1,NS5). **Section 5** indicates which packing method is to be used to pack the data field into the GRIB2 message. The contents of this array are modified by this routine. See the GRIB2 section outline below to determine which array elements the user needs to supply values for. (INPUT/OUTPUT)

<u>NS5</u> - The number of elements in the IS5( ) array. Since the data representation templates are of variable size, the value of this parameter depends upon which packing method is being used to pack the gridded data field. NS5=49 is sufficiently large enough for all of the **Section 5** data representation templates supported by this packer. (INPUT)

<u>IS6</u>(L) - Holds the data values of **Section 6** (the **Bit-map Section**) to be packed into the GRIB2 message (L=1,NS6). **Section 6** contains the optional, user-supplied bit-map (or bit mask) showing the locations of the primary missing values in the data field. The bit-map has a one-to-one correspondence to the data points in the data field. So, if the data field contains 1000 elements, then the bit-map will contain 1000 bits. A bit set to “1" indicates that the corresponding value in the data field is valid; a bit set to “0" indicates that the corresponding value in the data field is missing. **The contents of this array are modified by the packer.**

A bit-map is packed only when the **simple packing method** is being used. The **complex packing method** (with or without **second order spatial differences**) has a way of internally representing missing values. However, the user need to provide a bit-map to the packer even when the complex packing method is being used. For example, suppose a user has a data field from which the missing values have been removed and a bit-map that corresponds to that data field indicating the positions of missing values. Both the data field and the bit-map must be supplied to the packer regardless of the packing method that is being used. If the simple packing method is being used, then the bit-map and data field are packed just as they are. If the complex or complex with second order spatial differences packing method is being used, then the bit-map is used to insert the missing values back into the data field, and then the data field is packed (in this case the bit-map is not packed).

As another example, suppose the user has a data field with missing values in it, and there is no corresponding bit-map showing the locations of the missing values. If the **simple packing method** is being used, then the packer will generate the bit-map and remove the missing values from the data field before packing. If the **complex packing method** (with or without second order spatial differences) is being used, then the data field is packed just as it is (with the missing values embedded in it).

The processing of the bit-map is largely determined by the "IBITMAP" and "ICLEAN" calling arguments (described below). (INPUT/OUTPUT)

<u>NS6</u> - The number of elements in the IS6( ) array. NS6=6 is sufficient. (INPUT)

<u>IS7</u>(L) - Holds the data elements for **Section 7** (the **Data Section**) to be packed into the GRIB2 message (L=1,NS7). **Section** 7 contains the actual packed data field. The contents of this array are modified by this routine. See the GRIB2 section outline below to determine which array elements the user needs to supply values for. (INPUT/OUTPUT)

<u>NS7</u> - The number of elements in the IS7 array. NS7=8 is sufficient for all packing methods. (INPUT)

<u>IB</u>(L,M) - Contains the user-supplied bit-map (L=1,NX) (M=1,NY). The bit-map must have a one-to-one correspondence with the data field; that is, there must be a bit in the bit-map for each data value in the data field. A bit set to "1" in the bit-map indicates that the corresponding value in the data field is valid; a bit set to "0" in the bit-map indicates that the corresponding value in the data field is missing. When the user supplies a bit-map to the packer, the "IBITMAP" calling argument (see below) must be set to "1".

If the user does not supply a bit-map, and the data field being packed contains missing values (i.e., "ICLEAN" = 0, see below), and the simple packing method is being used to pack the data, then this routine will generate a bit-map in the IB( ) array. (INPUT/OUTPUT)

<u>IBITMAP</u> - A "flag" indicating whether or not the user is supplying a bit-map in the IB( ) array. A value of "1" indicates that a bit-map is being provided; a

value of "0" indicates that a bit-map is not being provided. If IBITMAP = 0 and PK_GRIB2 generates a bit map (see IB( ) above), IBITMAP is returned = 1. (INPUT/OUTPUT)

IPACK(L)- The array into which the GRIB2 message is packed and returned to the caller (L=1,ND5). (OUTPUT)

ND5 The dimension of the IPACK( ) array. ND5=(250+(NX*NY)+(NX* NY)/8 + the number of bytes of local use data) should be sufficient for most purposes for a single grid (see NEW below). (INPUT)

MISSP - The integer representation of the **primary missing value**. This is the value the packer will use when a data field comprised of integer values is being packed. When there are primary missing values and one of the complex packing methods is being used, then user must be sure to set IS5(23)=1. When IS5(23) = 0, MISSP is not used, but zero is inserted into IS5(24-27). (INPUT)

XMISSP - The floating point representation of the **primary missing value**. This is the value the packer will use when a data field comprised of floating point values is being packed. When there are primary missing values and one of the complex packing methods is being used, then user must be sure to set IS5(23)=1. When IS5(23) = 0, XMISSP is not used, but zero is inserted into IS5(24-27). (INPUT)

MISSS - The integer representation of the **secondary missing value**. This is the value the packer will use when a data field comprised of integer values is being packed. There must be a primary missing value specified in order to use a secondary missing value, but the primary value does not have to actually occur in the data. When there are primary and secondary missing values and the complex packing method is being used, the user must be sure to set IS5(23)=2. When IS5(23) = 0 or 1, XMISSP is not used, but its value is inserted into IS5(28-31). **Secondary missing values cannot be used with complex packing with second order spatial differences.** (INPUT)

XMISSS - The floating point representation of the **secondary missing value**. This is the value the packer will use when a data field comprised of real-type values is being packed. There must be a primary missing specified value in order to use a secondary missing value, but the primary value does not have to actually occur in the data. When there are both primary and secondary missing values to be considered and the complex packing method is being used, the user must be sure to set IS5(23)=2. When IS5(23) = 0 or 1, XMISSP is not used, but its value is inserted into IS5(28-31). **Secondary missing values cannot be used with complex packing with second order spatial differences.** (INPUT)

NEW - When packing only one gridded data field into a GRIB2 message, this parameter must always be set to "1". When packing multiple gridded fields into a single GRIB2 message, this parameter must be set to "1" while packing the first data grid and then set to "0" when packing the remaining data grids into the GRIB2 message. This flag indicates to the packer whether or not the grid currently being processed is the first grid to be packed into the GRIB2 message. (INPUT)

MINPK - This parameter applies only to the complex and complex with second order spatial differences packing methods. Its value represents the minimum size of the groups that the complex packing method can break the data down into. A recommended value is “14". Changes to this value will likely have an effect on the packing efficiency of this routine. Exactly what this affect is depends on the nature of the data. (INPUT)

ICLEAN - A flag indicating whether or not there are primary missing values in the data field to be packed by this routine. A value of “0" indicates that there are primary missing values embedded within the data field. A value of “1" indicates that there are not any primary missing values embedded in the data field. **This may be changed by PK_GRIB2.** (INPUT/OUTPUT).

JER(L,M)- Contains any diagnostic or error codes along with their severity levels generated in this routine (L=1,NDJER) (M=1,2). This error-handling scheme was developed to preserve all diagnostic information generated during execution. Since some error codes are non-fatal and offer information that is of diagnostic value, it is possible that a run of this GRIB2 encoder may generate several diagnostic codes. This array provides the user with a way of deducing a “trace back.” This error handling scheme works as follows:

The rows in the JER array represent individual error occurrences. The first column in the JER array represents the error code; the second column represents the severity of the error code.

There are three severity levels that can be assigned to an error code:

0 = Not a Problem
1 = Warning
2 = Fatal

An error with a severity level of "Warning" does not warrant the execution of the packer being aborted. An error with a severity level of "Fatal" results in the execution of the packer being halted even if the GRIB2 message has not been completely packed.

Each time the packer starts packing a new section of the GRIB2 message, it places a three digit section code representing the section being packed followed by a severity level of “0" into the first and second columns, respectively, of the next available row of the JER array. Section codes are 0 (Section 0), 100 (Section 1), 200 (Section 2), 300 (Section 3), 400 (Section 4), 500 (Section 5), 600 (Section 6), 700 (Section 7), and 800 (Section 8).

When an error is encountered while packing a GRIB2 message, the routine detecting the error will place the error code followed by its severity level into the first and second columns, respectively, of the next available row of the JER array.

For example, suppose a call to “pkbg” failed while packing Section 7 of a GRIB2 message because the “NBIT” calling argument did not have a value inclusively contained in the range of 0 to 32. The contents of the JER array upon being returned to the caller of the "pk_grib2" subroutine would appear as follows:

| Contents of JER | Diagnostic/Error Code | Severity |
|---|---|---|
| row 1 | 0 | 0 |
| row 2 | 100 | 0 |
| row 3 | 200 | 0 |

| | | |
|---|---|---|
| row 4 | 300 | 0 |
| row 5 | 400 | 0 |
| row 6 | 500 | 0 |
| row 7 | 600 | 0 |
| row 8 | 700 | 0 |
| row 9 | 3 | 2 |

This tells the user that all sections up to Section 7 were successfully packed. Note that the diagnostic/error code 0 corresponds to Section 0; 100 corresponds to Section 1; 200 corresponds to Section 2; 300 corresponds to Section 3; 400 corresponds to Section 4; 500 corresponds to Section 5; 600 corresponds to Section 6; and 700 corresponds to Section 7. Also note that since each of these section codes is followed by a severity level of 0, it means that the packing of the GRIB2 message has been successful UP TO THAT SECTION. The error code of “3" in row 9 is the error code generated by routine “pkbg” indicating the invalid value of the “NBIT” calling argument. The “2" in the severity column indicates that the error being returned is fatal and that the encoding of the GRIB2 message is being halted with return to the user.

The advantage to using this error handling scheme is that the caller can isolate where the problem occurred (in this example, Section 7). This problem would be very difficult to find if the user was given a single error code upon return from the pk_grib2 routine, especially since the pkbg utility is called throughout the entire encoder. This error handling scheme was created to give the user some type of error handling/traceback capability in lieu of the packer actually printing out diagnostic messages. However, if the user desires diagnostic output, see the notes corresponding to the “KFILDO” calling argument above. (OUTPUT)

<u>NDJER</u> - The number of rows in JER( ). It is recommended that this be set to at least “15". If this value is not set large enough and the JER array fills up, the last row of the JER array will be overwritten with an error code of “999" with a fatal severity level of “2". This will result in the loss of at least two diagnostic codes and their corresponding severity levels. The encoding of the GRIB2 message will be halted with return to the user. (INPUT)

KJER - The number of error/diagnostic messages contained within JER( ). Useful for testing for errors when program control is returned back to the calling routine. (OUTPUT)

OUTPUT:

Diagnostic messages will be written to Unit No. “KFILDO” when pk_grib2 has been compiled using the compile options as outlined above in the description of the “KFILDO” calling argument.

RESTRICTIONS:

Care must be taken when choosing primary and secondary missing values for a gridded data field. In general, the missing values should represent numbers which are several orders of magnitude larger than the actual data values. This is especially important when using the complex and complex with second order spatial differences packing methods. If the missing values are “too close” in magnitude to the real data values, it is possible that upon scaling and taking differences some real values will actually be treated as missing values leading to erroneous results.

NONSYSTEM ROUTINES USED:
See the user associated library.

LANGUAGE: FORTRAN 90

GRIB2 FORMAT:

Nine sections are defined for GRIB2. Sections in ( ) are optional.

| Section | Section Name | Section Contents |
|---|---|---|
| 0 | Indicator Section | “GRIB”, GRIB edition #, message length |
| 1 | Identification Section | Characteristics of all the processed data |
| (2) | (Local Use Section) | Additional items for local use |
| 3 | Grid Definition Section | Geometry of values |
| 4 | Product Definition Section | Description of following processed data |
| 5 | Data Representation | How the processed data are represented |
| 6 | Bit-map Section | Indicator of value being |

| | | |
|---|---|---|
| | | present/absent |
| 7 | Data Section | Binary data values |
| 8 | End Section | “7777" |

The contents of each section of the GRIB2 message, as well as number of octets (bytes) required to store each element in the section are detailed in the WMO document **FM 92-XII GRIB.** Arrays named IS0-IS7 are used to pass the required input/output values into and out of the packer. Each of these arrays corresponds to a section in the GRIB2 message, e.g. array IS0 corresponds to **Section 0**. The element number in each of these “IS” arrays corresponds to the beginning octet number where the data value is stored in the section. So, for example, a value that is stored starting in octet 5 of Section 5 would be placed into element 5 of the IS5 array. In the following outline, data that need to be provided by the user are indicated with an asterisk (*). All other data elements are supplied by the packer.

**Section 0:**

IS0(1) = GRIB name, stored in bytes 1-4

IS0(7) = Discipline - master table number, stored in byte 7 (*)

IS0(8) = GRIB edition number, stored in byte 8

IS0(9) = Total length of the GRIB message, stored in bytes 9-16

**Section 1:**

IS1(1) = Length of section, stored in bytes 1-4

IS1(5) = Number of section, “1", stored in byte 5 (*)

IS1(6) = ID of originating/generating center, stored in bytes 6-7 (*)

IS1(8) = ID of originating/generating sub-center, stored in bytes 8-9 (*)

IS1(10) = GRIB Master tables version number, stored in byte 10 (*)

IS1(11) = GRIB Local tables version number, stored in byte 11 (*)

IS1(12) = Significance of Reference Time, stored in byte 12 (*)

IS1(13) = Year (4 digits), stored in bytes 13-14 (*)

IS1(15) = Month, stored in byte 15 (*)

IS1(16) = Day, stored in byte 16 (*)

IS1(17) = Hour, stored in byte 17 (*)

IS1(18) = Minute, stored in byte 18 (*)

IS1(19) = Second, stored in byte 19 (*)

IS1(20) = Production status of processed data in message, stored in byte 20 (*)

IS1(21) = Type of processed data in message, stored in byte 21 (*)

**Section 2:**

See discussion below about **Local Use Data**.

**Section 3:**

IS3(1) = Length of section, stored in bytes 1-4
IS3(5) = Number of section, stored in byte 5 (*)
IS3(6) = Source of grid definition, stored in byte 6 (*)
IS3(7) = Number of data points, stored in bytes 7-10 (*)
IS3(11) = Number of octets for optional list of numbers defining number of points, stored in byte 11 (*)
IS3(12) = Interpretation of list of numbers defining number of points, stored in byte 12 (*)
IS3(13) = Grid Definition Template Number, stored in byte 13 (*)
IS3(15) - IS3(nn) = Grid Definition Template, stored in bytes 15-nn (*)

**Section 4:**

IS4(1) = Length of section, stored in bytes 1-4
IS4(5) = Number of section, stored in byte 5 (*)
IS4(6) = Number of coordinates values after the template, stored in bytes 6-7 (*)
IS4(8) = Product Definition Template Number, stored in bytes 8-9 (*)
IS4(10) - IS4(nn) = Product Definition Template, stored in bytes 10-nn (*)

**Section 5:**

IS5(1) = Length of section, stored in bytes 1-4
IS5(5) = Number of section, stored in byte 5 (*)
IS5(6) = Number of data points where one or more values are specified in Section 7 when a bit-map is present. Total number of data points when a bit-map is absent, stored in bytes 6-9
IS5(10) = Data Representation Template Number, stored in bytes 10-11 (*)
IS5(12) - IS5(nn) = Data Representation Template, stored in bytes 12-nn (*)

**Section 6:**

IS6(1) = Length of section, stored in bytes 1-4
IS6(5) = Number of section, stored in byte 5 (*)
IS6(6) = Bit-map indicator, stored in byte 6
IS6(7) - IS6(nn) = Bit map stored in bytes 7-nn

**Section 7:**

IS7(1) = Length of section, stored in bytes 1-4
IS7(5) = Number of section, stored in byte 5 (*)
IS7(6) - IS7(nn) = Data in a format described by the Data Template Representation. Template number given in octets 10-11 of section 5.

**Section 8:**

IS8(1) = “7777", the end of message indicator, stored in bytes 1-4

**Local Use Data (Section 2)**

GRIB2 provides the capability to preserve and pass along information about the gridded data that the GRIB2 format does not provide specific templates or code tables for. **Section 2** is provided to contain this local use data. GRIB2 does not specify any restrictions on the format of the data in Section 2, which gives much flexibility in determining how to store data in Section 2. The local use data processing scheme described below is the one employed by Version 1 of the MDL GRIB2 encoder software.

The MDL GRIB2 packer allows the user to store both integer and floating point groups of local use data. Since the amount of local use data can be large, the simple packing method is employed to compress the data in **Section 2**. The user must pick the decimal scale factor to use in compressing each group of local use data. Care must be taken when choosing the scale factor to optimize the data compression while not compromising the precision of the data. For instance, when storing ASCII text local use data, it is usually best to use a decimal scale factor of “0".

The local use data are supplied by the user through the arrays IDAT( ) and RDAT( ) which are calling arguments to the “pk_grib2" routine. Integer data are passed in using the IDAT( ) array while floating point data are passed in using the RDAT( ) array. The data must be supplied the following manner to ensure that they are interpreted and packed correctly:

| Array Element Number | Description of Content |
|---|---|
| 1 | Number of values in the first group of data (N1). |
| 2 | The decimal scale factor to use in packing the first group of local use data (must be a whole number). |
| 3 to (N1+2) | First group of local use data values. |
| N1+3 | Number of values in the second group of local use data (N2). |
| N1+4 | The decimal scale factor to use in packing the second group of local use data (must be a whole number). |
| (N1+5) to (N1+N2+4) | Second group of local use data |

| | |
|---|---|
| | values. |
| (K-1)*2+1+N1+N2+...+N(k-1) | Number of values in the Kth group of data (Nk) |
| (K-1)*2+2+N1+N2+...+N(k-1) | The decimal scale factor to use in packing the Kth group of data. |
| (K-1)*2+3+N1+N2+...+N(k-1)<br>to<br>(K-1)*2+N1+N2+...+N(k-1)+Nk) | The Kth group of local use data values. |
| (K-1)*2+1+N1+N2+...+Nk) | “0"<br>A value of “0" where the size of the group goes means that there are no more local use data supplied in this array. |

Note that this format applies to both the RDAT( ) and the IDAT( ) arrays. Also, if there are no local use data, then **element 1 of both arrays MUST have a value of “0".**

**Processing GRIB2 messages on Systems with Different Memory Architectures**

A few extra steps must be taken when using this software on a system that uses a “little-endian” memory architecture, where the low-order byte of a word is in the word’s starting address. The order of bytes in the message is at least implied to be high order first and specifically the WMO documentation reads concerning floating point, "The numbers are stored with the high order octet first. The sign bit will be the first bit of the first octet. The low order bit of the mantissa will be the last (eighth) bit of the fourth octet. This floating point representation has been chosen because it is in common use in modern computer hardware. Some computers use this representation with the order of the octets reversed. They will have to convert the representation, either by reversing the octets or by computing the floating point value directly..." PK_GRIB2 uses this "big endian" representation.

When packing a GRIB2 message on a “little-endian” system, it is essential that the bytes in the packed GRIB2 message be “swapped” to conform to “big-endian” standards before the message is sent. This should be done right after calling the “pk_grib2" routine to create the packed message. A routine, named “pk_swap”, is provided in the packer library to perform this byte swapping on the packed GRIB2 message. It takes as arguments the array containing the packed message and the number of elements in that array. The byte swapping is performed in-situ within the array.

The “pk_swap” routine is written in the C-language for greater efficiency. Exactly how this routine is linked into the user’s executable depends upon the linker that is being used and the language that the main routine is written in. When using a Fortran main routine, it is recommended that the user see the “man” pages supplied with the compiler/linker being used for information on how to call a “C” routine from a Fortran main routine.

The “pk_endian” function (also supplied in the packer library) can be called to determine the type of memory architecture of the system that the GRIB2 message is being created on. This function will return a value of “TRUE” on a “big-endian” system and a value of “FALSE” on a “little-endian” system. The following is a portion of a Fortran driver demonstrating how to use the “pk_swap” and “pk_endian” routines:

```
      PROGRAM PACKER
C
      LOGICAL BIG
      ...
C
      CALL PK_GRIB2(KFILDO, AIN,IAIN,NX,NY,IDAT,NIDAT,RDAT,NRDAT,
     1    IS0,NS0,IS1,NS1,IS3,NS3,IS4,NS4,IS5,NS5,IS6,NS6,
     2    IS7,NS7,IB,IBITMAP,IPACK,ND5,MISSP,XMISSP,MISSS,
     3    XMISSS,NEW,MINPK,ICLEAN,L3264B,JER,NDJER,KJER)
C
      IF(JER(KJER,2).EQ.2)THEN
         WRITE(*,15) JER(KJER,1)
 15      FORMAT(//,1X,'FATAL ERROR IN PK_GRIB2 ERROR CODE ',I5)
         STOP
      ENDIF
C
      BIG=PK_ENDIAN()
C
      IF(.NOT.BIG)THEN
         CALL PK_SWAP(IPACK,ND5)
      ENDIF
C
      ...
```

### Memory Management When Dealing with Large Data Grids

This routine makes extensive use of arrays, some of which can become very large and consume considerable amounts of memory. Normally, the user will declare in the main routine the arrays being passed into “pk_grib2". For small data grids, these arrays can be easily created on the **stack**. However, depending upon the platform that the packer is being run on, large grids of data such as those representing satellite data **may overflow stack memory** resulting in a segmentation violation and abnormal termination of the packer’s execution. A “work around” for this is to create the large arrays (such as IPACK( ) , AIN( , ) , IAIN( , ) , and IB( ))

on the **heap** through dynamic memory allocation since the **heap** typically offers much more storage than the **stack** does. The usual precautions that accompany dynamic memory allocation apply here. Namely, the user should free all dynamically allocated memory before the termination of the main program.

**Error Codes**

The following is a list of all of the possible diagnostic and error return codes that can be returned by this routine. An attempt has been made to give the numeric codes a meaningful appearance that will aid the user of this GRIB2 encoder in identifying which part of the GRIB2 software is generating the error condition. For example, errors encountered while packing Section 3 of the GRIB2 message will generally have a value ranging from 301 to 399 while those encountered while packing Section 4 of the GRIB2 message will generally have a value ranging from 401 to 499. Not all error codes represent “fatal” circumstances that require the termination of the pk_grib2 routine. Along with the actual value of an error code, severity information is also returned to the user indicating whether the error represents a “fatal” condition or is provided just for diagnostic purposes. **The bold routine name after each error code is the routine that the error code originates in.**

**0** = A good return **(all routines)**
**1** = Packing would overflow IPACK( ) **(pkbg.f)**
**2** = IPOS is not in the range of 1 to L3264B **(pkbg.f) (pk_s7.f, pk_c7.f, length.f)**
**3** = NBIT is not in the range of 0 to 32 **(pkbg.f, pkc7.f, pks7.f)**
**4** = NBIT is equal to “0", but NVALUE is not equal to “0" **(pkbg.f)**
**5** = LOC2 points to a word in IPACK( ) before the word pointed to by LOC1 **(length.f)**
**6** = LOC2 and LOC1 point to the same word, but IPOS2 points to a bit position before IPOS1 **(length.f)**
**8** = Invalid option of data type in IS5(21) **(pk_grib2.f)**
**101** = IS1(5) is not equal to value of "1" signifying Section 1 **(pk_sect1.f)**
**102** = IS1( ) has not been dimensioned large enough **(pk_sect1.f)**
**202** = The IDAT( ) or RDAT( ) array was not dimensioned large enough to contain the local use data. **(pk_sect2.f)**
**301** = Section 3 exists and IS3(5) does not indicate Section 3 **(pk_sect3.f)**
**302** = IS3( ) has not been dimensioned large enough to contain the entire template**(pk_azimuth.f, pk_cylinder.f, pk_equator.f, pk_lambert.f, pk_mercator.f, pk_orthographic.f, pk_polster.f, pk_sect3.f)**
**303** = Map projection in IS3(13) is not supported **(pk_sect3.f)**

**304** = IS3(13) does not indicate a proper map projection; incorrectly called subroutine **(pk_azimuth.f, pk_cylinder.f, pk_equator.f, pk_lambert.f, pk_mercator.f, pk_orthographic.f, pk_polster.f)**
**305** = Section 2 exists but Section 3 does not **(pk_grib2.f)**
**310** = Unsupported shape of earth code in IS3(15) **(earth.f)**
**401** = IS4(5) does not indicate Section 4 **(pk_sect4.f)**
**402** = IS4( ) has not been dimensioned large enough to contain the entire template indicated by the code in IS4(6). Returned by routines like **(pk_temp40.f, pk_sect41.f, pk_sect42.f, pk_sect420.f, pk_sect430.f, pk_sect48.f, pk_sect4.f)**
**403** = IS4(8) does not contain a supported template number **(pk_sect4.f)**
**501** = IS5(5) does not indicate Section 5 **(pk_sect5.f)**
**502** = IS5( ) has not been dimensioned large enough to contain Section 5 **(pk_sect5.f)**
**508** = Unsupported packing type **(prepr.f)**
**601** = IS6(5) does not indicate Section 6 **(pk_sect6.f)**
**602** = IS6( ) has not been dimensioned large enough to contain Section 6 **(pk_sect6.f)**
**605** = Not enough space in IPACK( ) to contain the packed GRIB2 message **(pk_bmap.f)**
**701** = IS7(5) does not indicate Section 7 **(pk_sect7.f)**
**702** = IS7( ) has not been dimensioned large enough to contain Section 7 **(pk_sect7.f)**
**703** = Unsupported type of packing **(pk_sect7.f)**
**705** = ND5 is not large enough to accommodate the bits necessary to pack the values starting at the location indicated by LOCN and IPOS **(pk_c7.f, pk_s7.f)**
**706** = Value will not pack in 30 bits. **(pack_gp.f; pk_smple.f)**
**711** = LBIT contains an incorrect value **(pk_cmplx.f)**
**712** = Incorrect splitting method **(pk_cmplx.f)**
**713** = Unrecognized missing value flag in IS5(23) **(pk_cmplx.f)**
**714** = Problem in REDUCE. Results still ok, but more space may be required and extra processing required. **(reduce.f)**
**715** = NGP not large enough. Results still ok, but additional processing time required. **(reduce.f)**
**716** = MINPK locally increased. Results still ok **(pack_gp.f)**
**717** = INC set = 1 locally. Results still ok **(pack_gp.f)**
**901** = An invalid combination of ICLEAN, IS5(10) (packing method), and (Missing values) was specified. See external documentation. **(check_int.f, check_flt.f)**
**902** = There are no "good" values in the grid and the bit-map indicates that. Not treated as a “fatal” error in prepr **(prep_noval.f)**
**903** = There are no values in the grid and the bit-map indicates that there should be. Treated as fatal in prepr **(prep_noval.f)**

**904** = There are no values in the grid, and there is no bit-map. Treated as a “fatal” error in prepr **(prep_noval.f)**
**905** = Type of data in IS5(21) not correct **(prepr.f)**
**906** = Simple packing, no missing values in array, bit map provided, and NVAL = NXY. Unusual; notify the user **(pack_opt.f)**
**907** = Simple packing, no missing values in array, no bit map provided, but NVAL is not equal NXY, unrecoverable error **(pack_opt.f)**
**908** = IS5(23) set = 0 consistent with ICLEAN = 1 and not simple packing **(pack_opt.f)**
**909** = IS5(23) set = 0 and ICLEAN set = 1 because there are no missing values in the field **(check_int.f; check_flt.f, prepr.f)**
**910** = IS5(23) set = 1 because there are only primary missing values in the field **(check_int.f; check_flt.f, pack_opt.f)**
**911** = IS5(10) set = 2 to indicate complex without 2nd order differences because it is more efficient **(pack_opt.f)**
**912** = IS5(23) set = 2 to indicate secondary missing values **(pack_opt.f)**
**913** = Bit map being generated because input ICLEAN = 0 and IBITMAP = 0 for simple packing **(flt_map.f, int_map.f)**
**914** = Bit map being incorporated into the data for non-simple packing **(flt_map.f, int_map.f)**
**915** = IS5(10) set = 2 because there are secondary missing values and 2nd order differences not supported **(pack_opt.f)**
**916** = MISSS = MISSP, when IS5(23) = 2. IS5(23) set = 1. **(check_flt.f, check_int.f)**
**920** = A value larger than what can be packed into four bytes has been encountered. **(pk_missp.f, pk_nomiss.f, pk_int, pk_flt)**
**999** = The JER( ) array is full **(pk_trace.f)**
**1002**= IS0( ) has not been dimensioned large enough **(pk_sect0.f )**

**Version Control:**

This Version 1.1 has been modified from Version 1.0 only 1) to add more diagnostic messages, ) to improve readability and comments, and 3) to correct any deficiencies found through rigorous testing. GRIB offers many options, only a portion of which are supported by PK_GRIB; even so, the number of combinations is large and some errors could still remain.